

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-123698

(43) 公開日 平成8年(1996)5月17日

(51) Int.Cl.⁶

G 0 6 F 9/46

識別記号

3 1 3 Z 7737-5B

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数 2 F D (全 8 頁)

(21) 出願番号 特願平6-286128

(22) 出願日 平成6年(1994)10月27日

(71) 出願人 000002185

ソニー株式会社

東京都品川区北品川6丁目7番35号

(72) 発明者 里見 清

神奈川県横浜市港南区最戸2-18-3

メゾン上大岡101

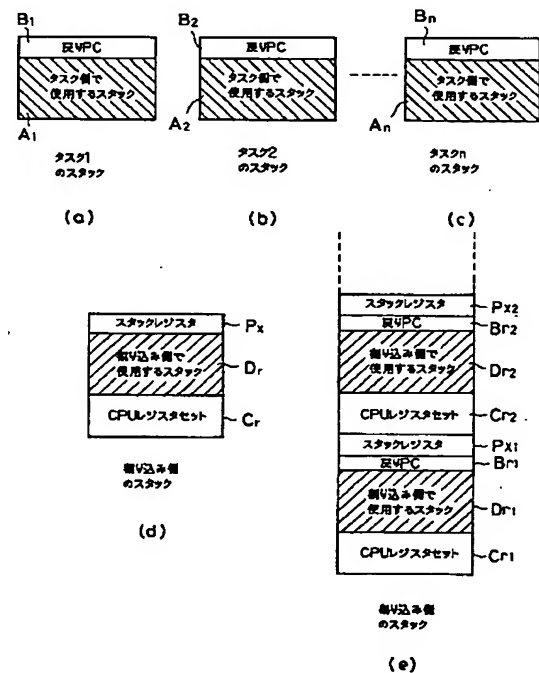
(74) 代理人 弁理士 藤 篤夫 (外1名)

(54) 【発明の名称】 プログラム構造

(57) 【要約】

【目的】 マルチタスクシステムにおいて、割り込み処理が使用するスタックを共通化しRAMサイズを節約する。

【構成】 割り込み処理が発生したときに、戻りPカウンタBにスタックAの中断ポイント、及びワークエリアPxにいずれのスタックが退避したかを記憶したのちにスタックポインタを割り込み処理側に切替え、上記割り込み処理が終了したときに戻りPカウンタB、及びワークエリアPxに記憶されたデータに基づきCPUレジスタセットCに退避したスタックポインタを復帰させるようにする。



【特許請求の範囲】

【請求項1】 マルチタスクシステムを実行するn個のスタックと、

上記n個のスタック個々に対応して構築され、各スタックの中断ポイントを記憶するプログラムカウンタと、

上記n個の各スタックに対して共有されて機能する割り込み処理スタックと、

割り込み処理が発生したときに実行中のスタックが退避されるCPUレジスタと、

上記割り込み処理によって上記n個のスタックのいずれのスタックが上記CPUレジスタに退避したかを記憶するスタックレジスタにより構成され、

上記割り込み処理が発生したときに、プログラムカウンタにスタックの中断ポイント、及び上記スタックレジスタにいずれのスタックが退避したかを記憶したのちにスタックポイントを割り込み処理側に切替え、上記割り込み処理が終了したときに上記プログラムカウンタ、及び上記スタックレジスタに記憶されたデータに基づき上記CPUレジスタに退避したスタックを復帰させるようにしたことを特徴とするプログラム構造。

【請求項2】 上記割り込み処理スタック、及びCPUレジスタ、及びスタックレジスタをそれぞれ複数構成し多重割り込みに対応するようにしたことを特徴とする請求項1に記載のプログラム構造。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、マルチタスクシステムを実行する場合に、メモリの容量を節約するプログラム構造に関するものである。

【0002】

【従来の技術】 最近では、コンピュータによる仕事の単位であるタスクを、一度に2以上こなすことができるマルチタスクシステムが一般的になってきている。このマルチタスクシステムを用いることによって、複数のタスクを終了させることなく切替えて実行することができるようになる。

【0003】 図4はマルチタスクシステムにおける、RAM(Random Access Memory)の使用状況の一例を模式的に示す図である。これらの図は、各タスクが使用するメモリ領域(以下、スタックという)毎に示されている。すなわち、RAM上にはマルチタスクシステムによって実行される複数のタスク1、タスク2・・・タスクnの複数のスタックが構築されることとなる。

【0004】 各スタックは、タスクが使用するスタックA(A₁～A_n)、戻りPカウンタ(Program Counter)B(B₁～B_n)、CPUレジスタセットC(C₁～C_n)、割り込み処理が使用するスタックD(D₁～D_n)などによって構成されている。タスクが使用するスタックAは現在実行されているタスクが動作するエリア、戻りPカウンタBは現在実行されているタスクの処

理ステップなどの処理ポイントを随時記憶するエリアである。また、CPUレジスタセットCはスタックDにおいて割り込み処理が実行されるときに現在実行中のタスクが退避するエリアである。

【0005】 上記した割り込み処理とはタスク(通常処理)が実行されているときに、このタスクを一時的に中断して、例えばタイマ等の制御により所定の時間毎に実行される処理や、リモートコマンド等によって実行される処理である。そして割り込み処理が使用するスタックD₁、D₂・・・D_nではすべて同じ内容の処理が実行される。例えば図4(a)に示したタスク1が実行されているときに、割り込み処理が発生した場合は、タスク1が実行されているスタックA₁とともに構築されている割り込み処理用のスタックD₁において実行されることとなる。また同じように同図(b)に示されているタスク2が実行されているときに割り込み処理が実行される場合は、タスク2が実行されているスタックA₂とともに構築されている割り込み処理用のスタックD₂において実行されることとなる。

【0006】 割り込み処理が行われる場合は、スタックA₁で実行されているタスク1は一時中断してCPUレジスタC₁に退避することとなり、このとき戻りPカウンタB₁に中断ポイントが記憶される。そして割り込み処理が終了すると戻りPカウンタB₁に記憶された中断ポイントに戻って処理を再開する。このように、各スタックに通常処理、及び割り込み処理用のスタックを構築することにより、マルチタスクシステムによって複数のタスクを実行中に所定の割り込み処理を行うことができるようになっている。

【0007】

【発明が解決しようとする課題】 しかしながら、図4に示したスタック構造では、タスク側で使用するスタックA、割り込み処理側で使用するスタックD及びCPUレジスタセットCを各タスク毎に確保しているので、タスク数が増加する場合はそれにもとないRAMの容量も増加することとなり、大容量のメモリが必要になってくる。また、割り込み処理側で使用するスタックサイズが、割り込み処理毎に変更されると、すべてのスタックサイズに影響をあたえるため、トラブルが発生しやすくなるという問題点がある。

【0008】

【課題を解決するための手段】 本発明はこのような問題点を解決するためになされたもので、マルチタスクシステムを実行するn個のスタックと、上記n個のスタック個々に対応して構築され、各スタックの中断ポイントを記憶するプログラムカウンタと、上記n個の各スタックに対して共有されて機能する割り込み処理スタックと、割り込み処理が発生したときに実行中のスタックが退避されるCPUレジスタと、上記割り込み処理によって上記n個のスタックのいずれのスタックが上記CPUレジ

スタに退避したかを記憶するスタックレジスタにより構成され、上記割り込み処理が発生したときに、プログラムカウンタにスタックの中断ポイント、及び上記スタックレジスタにいずれのスタックが退避したかを記憶したのちにスタックポインタを割り込み処理側に切替え、上記割り込み処理が終了したときに上記プログラムカウンタ、及び上記スタックレジスタに記憶されたデータに基づき上記CPUレジスタに退避したスタックを復帰させてスタック側の処理を再開させるようにする。また、上記割り込み処理スタック、及びCPUレジスタ、及びスタックレジスタをそれぞれ複数構成し多重割り込みに対応するように構築する。

【0009】

【作用】本発明によれば、割り込み処理が使用するメモリの容量を削減して節約することができるようになる。

【0010】

【実施例】以下、本発明の実施例を説明する。図1

(a) (b) (c) (d) (e) は本実施例のプログラム構造を模式的に示す図であり、図4と同一部分は同一符号を付して説明を省略する。本実施例のプログラム構造のタスク1～nのスタックは、同図(a) (b)

(c) に示されているように、タスク側で使用するスタックA₁～A_n、及び戻りPカウンタB₁～B_nで構成され、同図(d)に示されているように割り込み処理側のスタックDr、及びCPUレジスタセットCrはスタック1～nとは別に構築されている。そしてタスク1～nに割り込み処理が発生した場合は割り込み処理側のスタックDr、及びCPUレジスタセットCrを共有して処理を行うこととなる。

【0011】つまり、図4で説明した従来のプログラム構造と比較して、各スタックに対応して構築された割り込み処理側のスタックDi～Dn、及びCPUレジスタセットCi～Cnが占める容量が削減されることとなる。また、本発明では割り込み処理側のスタックDrとともに、スタックレジスタPxが設けられ、どのタスクがCPUレジスタセットCrに退避したか(スタックポインタ)を記憶するようになされている。

【0012】例えば同図(a)に示されているタスク1が実行されているときに割り込み処理を行う場合は、タスク1で実行されているCPU内容を同図(d)に示されているCPUレジスタセットCrに退避させる。このとき、スタックレジスタPxにはCPUレジスタセットCrに現在タスク1が退避していることが記憶され、さらに、戻りPカウンタBiにタスクの中断ポイントを記憶する。そして、割り込みが終了するとスタックレジスタPx及び戻りPカウンタBiがタスク1の所定のポイントから処理が再開されることとなる。

【0013】図2は実施例のマルチタスクシステムにおける割り込み処理を示すフローチャートである。なお、この図に示されているフローチャートは複数の割り込み

処理を同時に受け付ける多重割り込みを禁止した場合の一例を示している。まず、ステップS001で割り込みが禁止とされ、これによって多重割り込みが禁止されることとなる。そして、現在実行中のタスクのスタックポインタをスタックレジスタPxに退避させて(S002)、スタックポインタを割り込み側のスタックに切替え(S003)、さらに、図1(d)に示したCPUレジスタセットCrに現在まで実行していたタスクのCPUレジスタを退避させる(S004)。なお、実行が中断されたタスクの中断ポイントはそのタスクの戻りPカウンタB(B₁～B_nのいずれか)に記憶されている。このように、ステップS002～S004の処理を経た後にステップS005において割り込み処理が行われることとなる。

【0014】割り込み処理が行われた後は、まずステップS004でCPUレジスタセットCrに退避させたCPUレジスタを復帰させる(S006)。そしてさらに、ステップS002でスタックレジスタに退避させたスタックポインタを復帰させて割り込み処理を終了させ(S007)、他の割り込み処理を許可するようにする(S008)。割り込み処理が行われたのちに再開されるタスクの開始ポイントは、元のスタックの戻りPカウンタBに記録されたポイントとされ、このポイントから割り込み処理によって中断されたタスクが再開されることとなる(S009)。

【0015】また図3のフローチャートに示されているように、割り込み処理を行っている最中でも他の割り込み処理を受け付ける多重割り込み処理に対応することもできる。この場合図1(e)に示したようにCPUレジスタセットCr(Cr₁、Cr₂・・・)、スタックDr(Dr₁、Dr₂・・・)、スタックレジスタPx(Px₁、Px₂・・・)を割り込み処理の数に対応して複数構成する。

【0016】多重割り込み処理を行う場合は、割り込み処理がタスク側の処理を実行中に発生したか、又は割り込み処理中に発生したかをチェックするために、割り込みカウンタFを設けて現在実行中の処理を判別するようにする。すなわち、割り込みカウンタFが例えば『0』である場合はタスク側の処理を実行中、また割り込みカウンタFが『0』以外の『1』、『2』、『3』・・・である場合は割り込み処理を実行中にその数値に対応した数の割り込み処理が発生したこととなる。

【0017】まず、初期設定として割り込みカウンタFに『0』を設定する(S101)。そして割り込み処理が発生すると(S102)、ここでは一旦割り込み処理を禁止して(S103)、割り込みカウンタの判別をおこなう(S104)。このとき、割り込みカウンタFが『0』である場合はタスク処理への割り込みを行うためにステップS105に進み、割り込みカウンタFが『0』以外である場合は既に実行されている割り込み処理に対して割り込みを行うためにステップS105を迂回してステップS106に進む。

【0018】ステップS105では先程図2に示したステップS002～S003と同様に、タスク側のスタックポインタを退避して、スタックポインタを割り込みスタック側に切替え、さらにステップS106で割り込みカウンタFのインクリメント処理がなされる。そして実行していたタスクのCPUレジスタを退避した後に（S107）割り込みが許可され（S108）、第Fの割り込み処理が行われることとなる（S109）。

【0019】ステップS109で割り込み処理が行われているときにステップS110で割り込み処理の発生が検出されると、ステップS111に進み第Fの割り込み処理のレジスタを退避させ、再びステップS103にもどり所定の処理がなされることとなる。つまり、多重割り込みの場合、その割り込みの回数だけステップS106で割り込みカウンタがインクリメントされるようになる。したがって、割り込み処理が発生する間はステップS103～S111の処理を繰り返して第Fの割り込み処理が行われる。

【0020】また、ステップS110で割り込み処理がないと判別された場合はステップS112に進み、ステップS103～S111で行われている第Fの割り込み処理が終了したか否かを判別する。ここで第Fの割り込み処理が終了していないと判別されるとステップS109に戻り第Fの割り込み処理が続けられる。また第Fの割り込み処理が終了したと判別されるとステップS113に進み割り込みが禁止される。そして割り込みカウンタFがデクリメントされ（S114）、割り込みカウンタFが『0』か否かを判別する（S115）。ここで割り込みカウンタFが『0』でない場合は第Fの割り込み処理のレジスタ及び戻りPカウンタB、すなわち今まで実行していた割り込み処理の前の割り込み処理についてのレジスタ及び戻りPカウンタを復帰させ（S116）、ステップS108にもどり第Fの割り込み処理が行われる。つまり、割り込みカウンタFをデクリメントすることによって、現在までに割り込んだ処理を順次さかのぼって実行していくこととなる。

【0021】また、割り込みカウンタFが『0』である場合は、割り込み処理がすべて終了したこととなり、タスク側のスタックのレジスタを復帰（S117）、さらにタスク側のスタックポインタを復帰させ（S118）、割り込みを許可した後にタスク側の処理へと移行するようにする。

【0022】

【発明の効果】以上、説明したように本発明はマルチタスクシステムにおいてタスクが使用するスタックを複数構成しても、割り込み処理が利用するスタックを共通化しているので、メモリの容量を（タスク数－1）×（退避するCPUレジスタセット＋割り込み側で使用するスタック数）だけ節約することができるようになる。したがって、小容量のメモリでマルチタスクシステムを稼働することが可能となる。また、多重割り込みが発生した場合でも、その多重割り込み処理用のスタックは割り込み処理スタック上に構築されるので、メモリ節約を実現することができるという利点がある。

【図面の簡単な説明】

【図1】本発明の実施例のスタック構造を模式的に示す図である。

【図2】実施例の割り込み処理の流れを説明するフローチャートである。

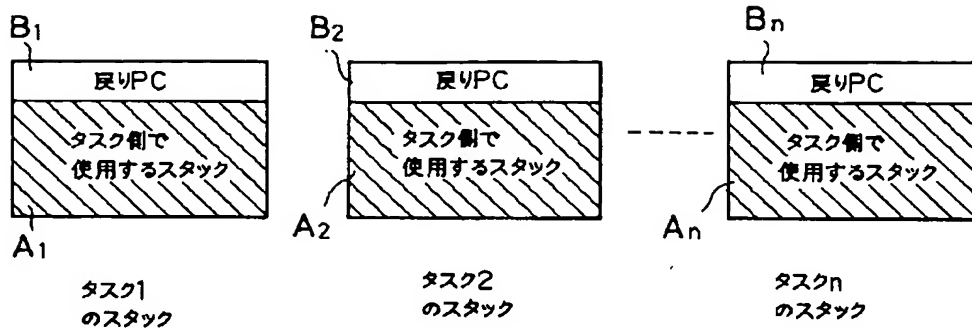
【図3】実施例の多重割り込み処理の流れを説明するフローチャートである。

【図4】従来のマルチタスクシステムにおけるスタック構造を模式的に示す図である。

【符号の説明】

- A タスク側で使用するスタック
- B 戻りPカウンタ
- C CPUレジスタセット
- D 割り込み処理で使用するスタック
- Px スタックレジスタ

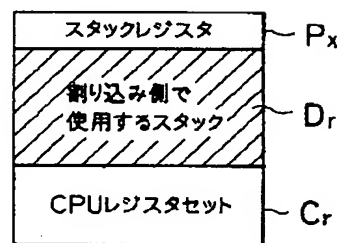
【図1】



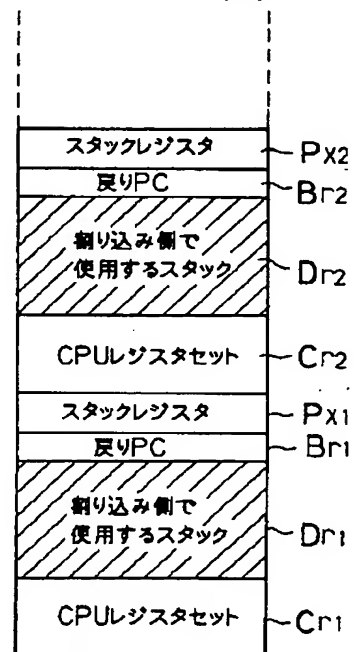
(a)

(b)

(c)

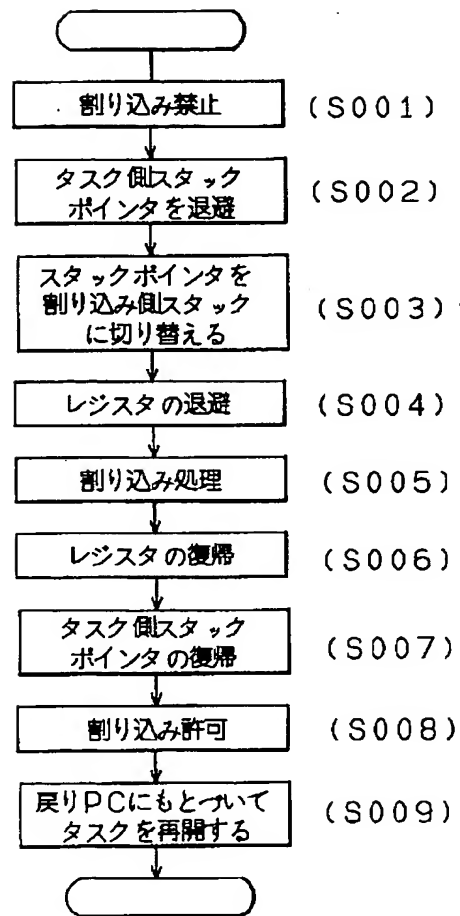
割り込み側
のスタック

(d)

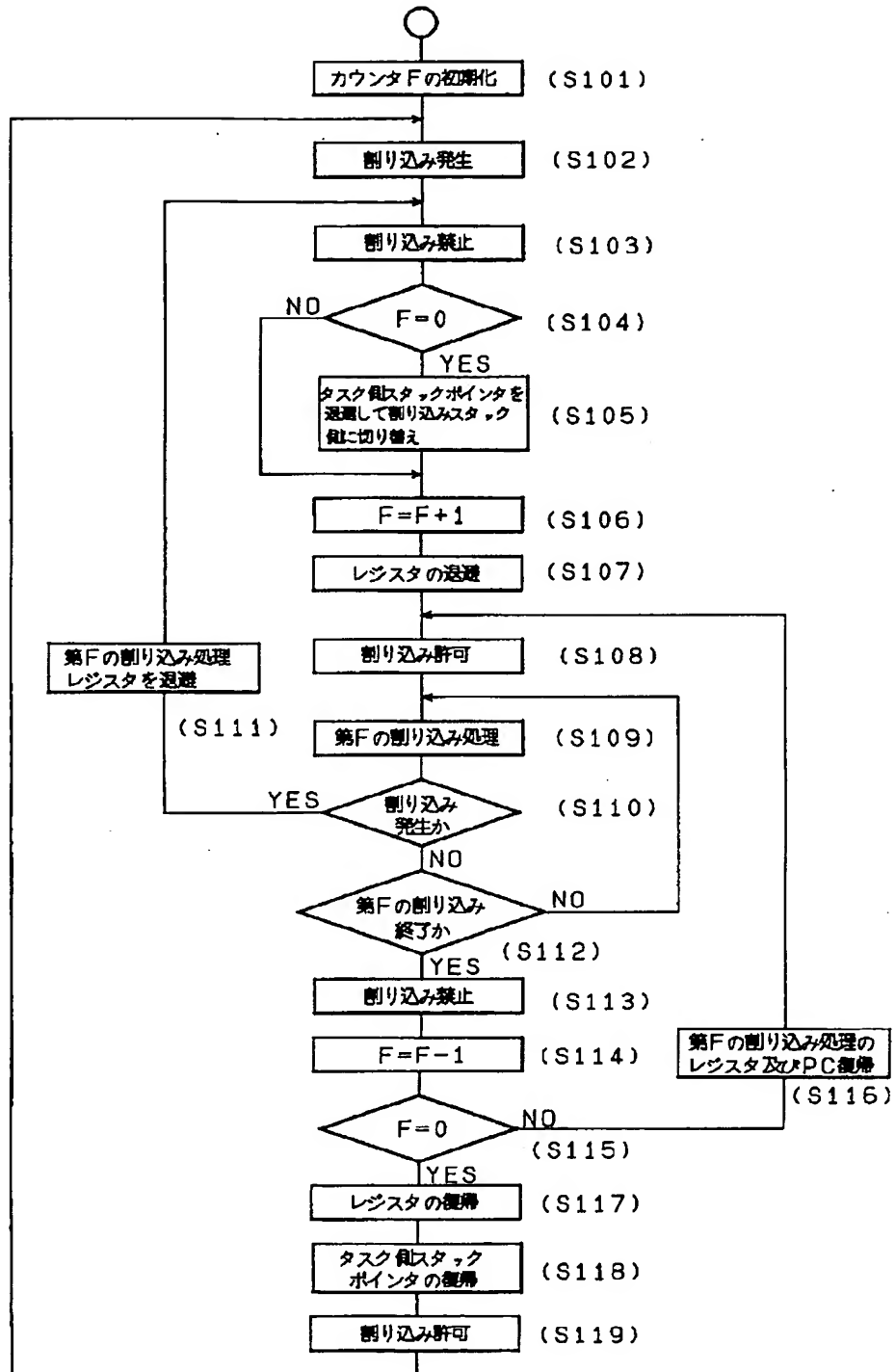
割り込み側
のスタック

(e)

【図 2】



【図3】



【図4】

